**LUCRARE DE ATESTAT**

Prezentarea algoritmului fill

Absolvent: Profesor coordonator:

Pop Mihai-Gabriel Mureșan Vasile Ciprian

Bistrița 2021

Cuprins

[INTRODUCERE 3](#_Toc72263397)

[MODALITATE DE IMPLEMENTARE 6](#_Toc72263398)

[MANUAL DE UTILIZARE 7](#_Toc72263399)

[COD SURSĂ 9](#_Toc72263400)

[BIBLIOGRAFIE 12](#_Toc72263401)

# INTRODUCERE

Aplicația **Algoritmul fill** este destinată elevilor de clasa a X-a care studiază capitolul Recursivitate, profesorilor care predau la clasă și programatorilor începători care doresc să înțeleagă algoritmul fill. Aplicația conține aspecte teoretice și un exemplu grafic prin care utilizatorul să înțeleagă mai bine modul în care funcționează algoritmul.

Aplicația este organizată pe 3 pagini cu subzone:

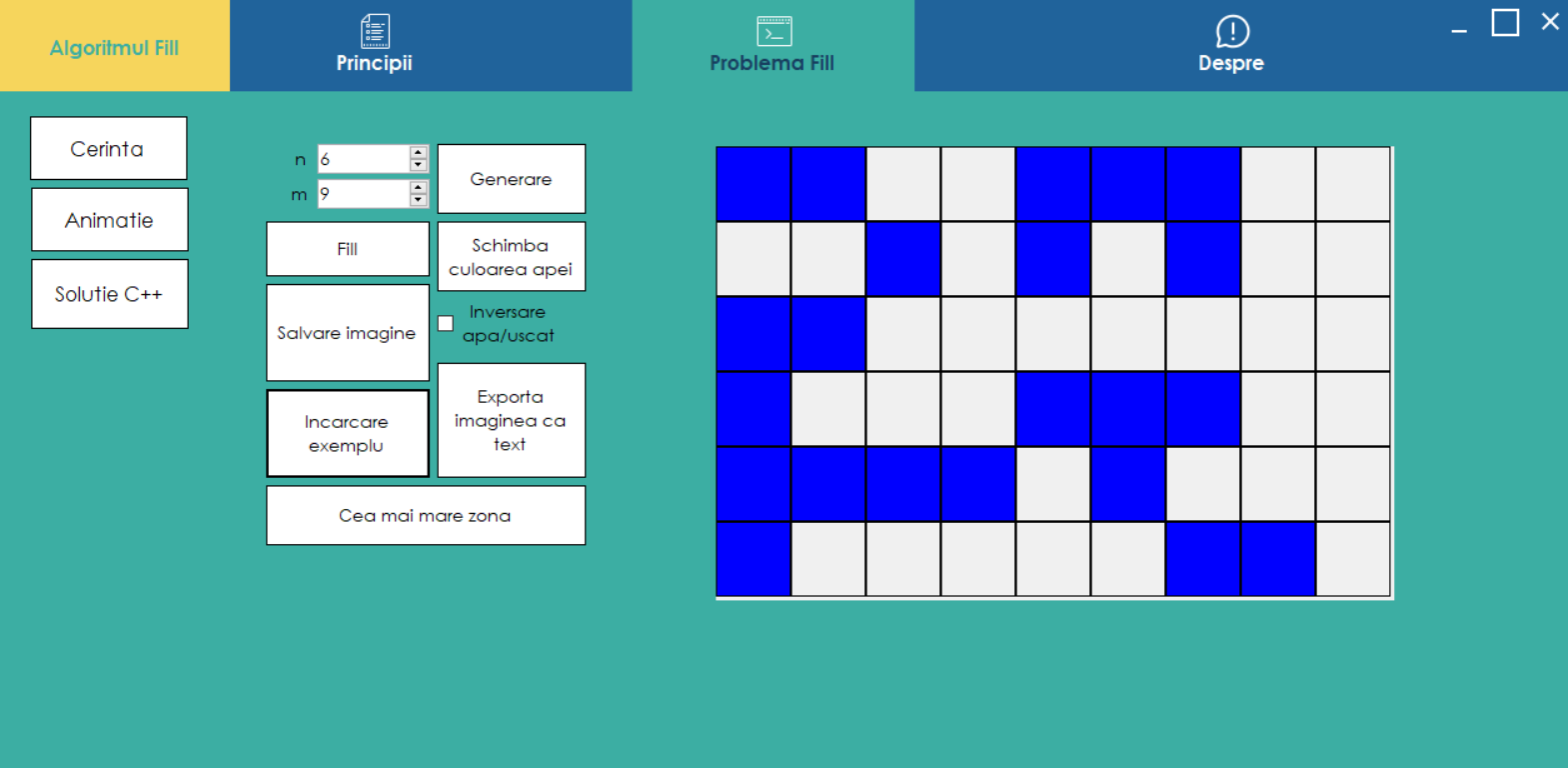
1. Principii: sunt explicate conceptele generale asupra acestui algoritm de umplere
2. Problema fill: este prezentată o problemă de pe pbinfo.ro și rezolvată. Această pagină mai conține:
   1. Cerința problemei
   2. Animația
   3. Sursa în limbajul C++
3. Despre: informații despre proiect

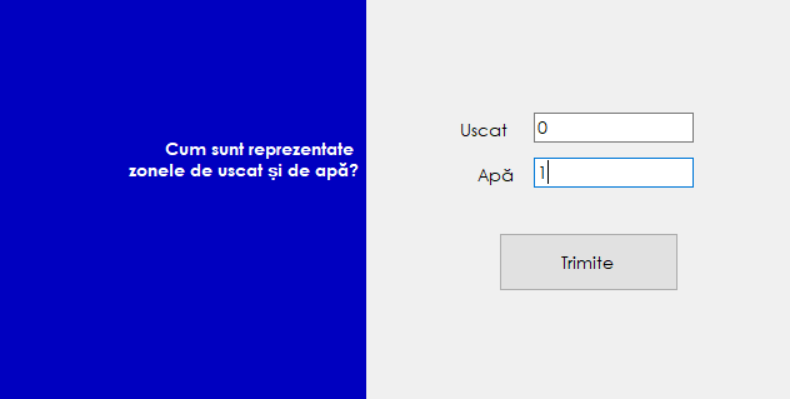
Manualul de utilizare nu este prezent în aplicație, el fiind integrat în această lucrare, într-un capitol ulterior.

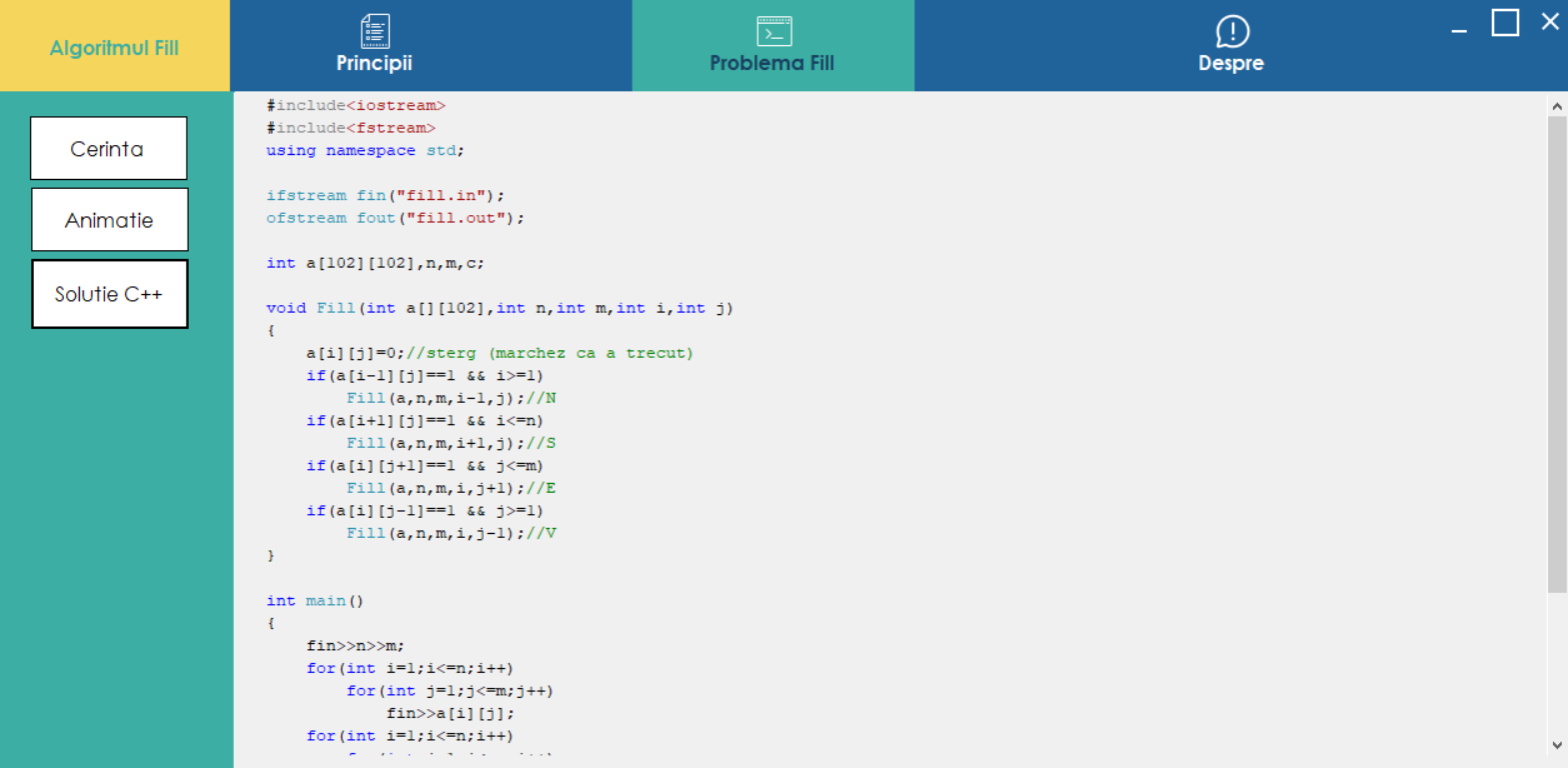
Aplicația este redimensionabilă, având două moduri, modul full screen și modul normal.

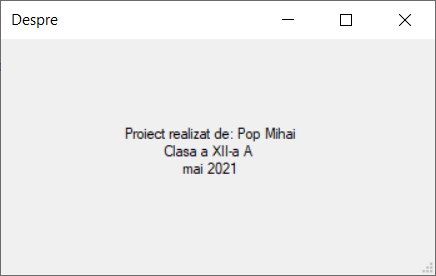












# MODALITATE DE IMPLEMENTARE

Pentru a implementa animația am folosit o matrice de butoane. Butoanele se generează automat în funcție de datele de intrare pe care le introduce utilizatorul. Se pot introduce în două moduri aceste date:

1. Manual: utilizatorul introduce în controale dimensiunea matricei, iar pe urmă pentru fiecare obstacol se apasă pe fiecare buton ce reprezintă o zonă inaccesibilă.
2. Automat: Cu ajutorul unui fișier de tip .in sau .txt.

Pentru animație am folosit un timer, iar în timp ce se verifică dacă un bloc este accesibil sau nu, colorez pentru o scurtă durată căsuțele prin care atrecut deja sau sunt inaccesibile cu roșu.

Pentru a construi tab-urile din aplicație am folosit usercontrols. Pentru tabul de principii am folosit un usercontrol care include un web browser pentru a deschide o pagină HTML. Pentru tabul *Problema Fill* am împărțit zona cu ajutorul a două panel-uri. Primul panel include butoanele prin care user-ul alege ce dorește să vizualizeze(cerința, animația sau soluția C++) și un panel în care am inclus cele 3 usercontrol, câte unul pentru fiecare buton. În momentul în care user-ul apasă un buton din panel-ul din stânga, usercontrol-ul corespunzător se afișează în mod fill în panel-ul respectiv, la celelalte două setându-se vizibilitatea ca fiind falsă pentru a le ascunde mai ușor. Soluția în C++ a fost introdusă într-un fișier .rtf pentru a păstra codul cromatic pentru fiecare instrucțiune. În proiectul C# am adăugat pe urma într-un usercontrol un richtextbox în care am citit fișierul respectiv.

# MANUAL DE UTILIZARE

1. Fila *Principii*

În această filă utilizatorului i se afișează informații generale despre algoritmul de umplere.

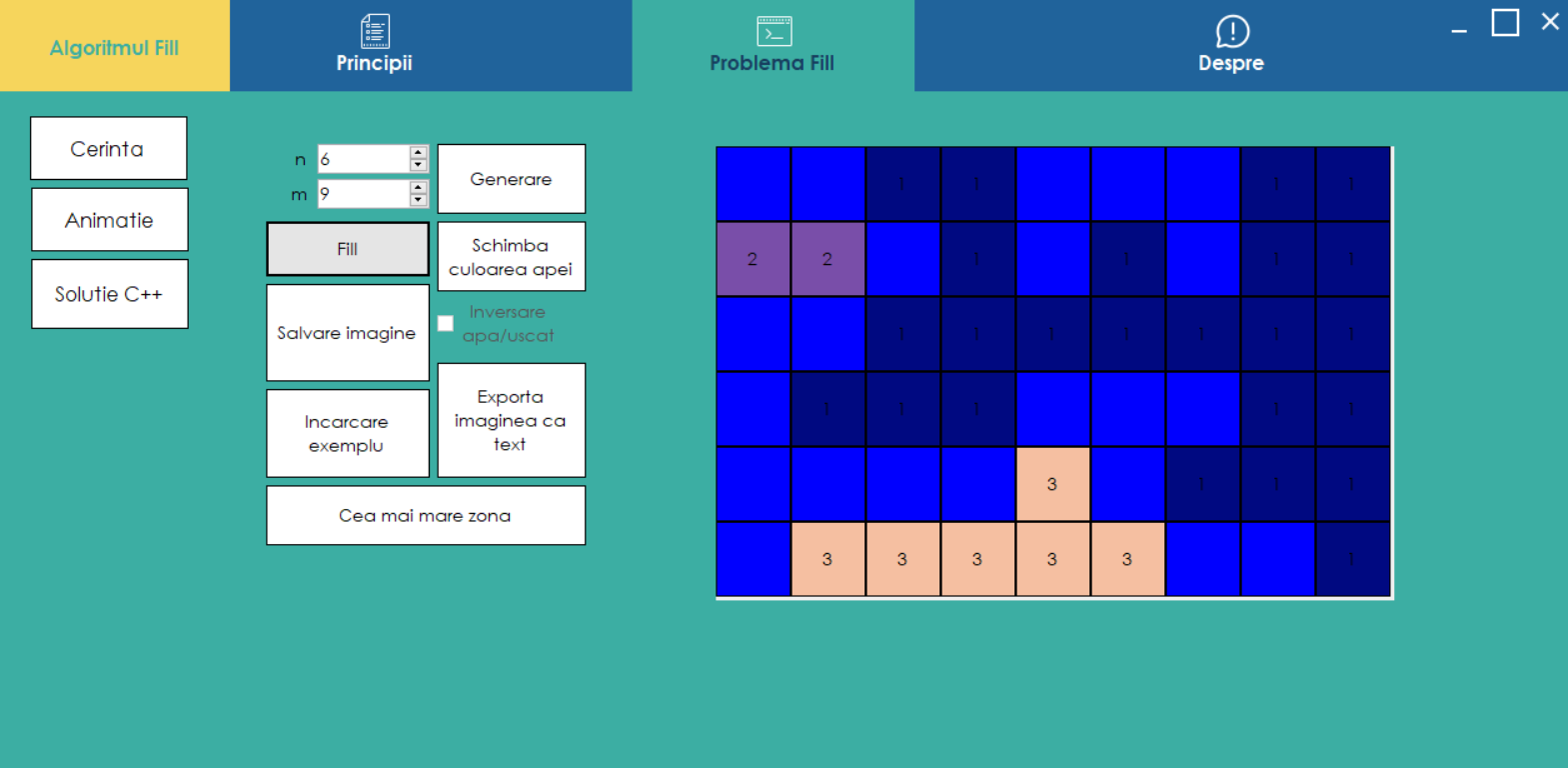
Graphical user interface

Description automatically generated

1. Fila *Problema Fill*

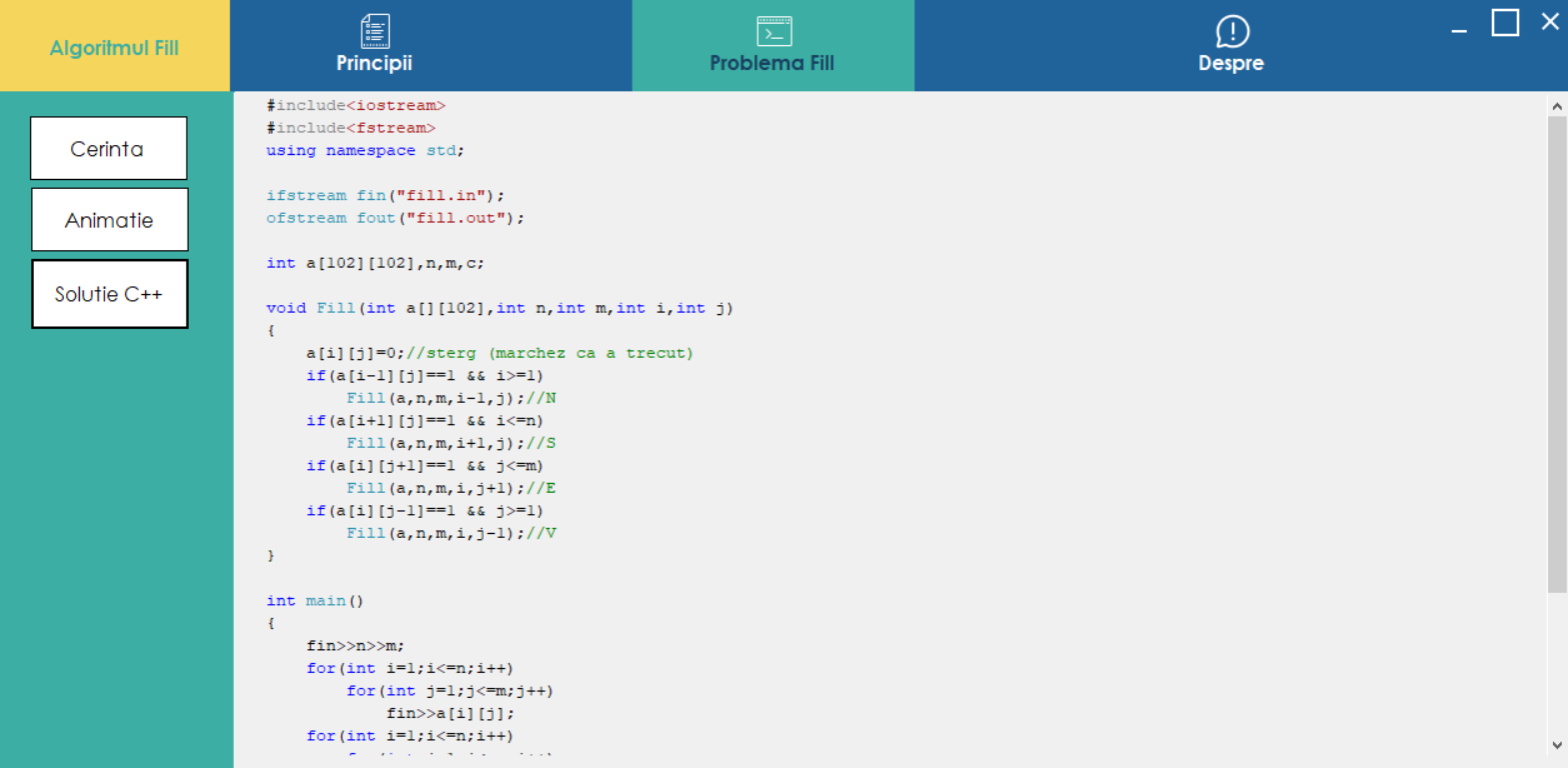
În această filă utilizatorul este întâmpinat de 3 butoane laterale: *Cerinta, Animatie* și *Solutia C++*. La apăsarea primului buton, utilizatorul este întâmpinat de cerința de pe pbinfo.ro a problemei Fill. Prin apăsarea butonului *Animatie*, utilizatorului i se afișează mai multe butoane în partea din stânga. Aceste butoane sunt:

1. Generare: generează matricea în funcție de ordinele de mărime introduse în controalele cu label n, respectiv m;
2. Fill: buton ce generează animația de umplere și de execuție a algoritmului fill;
3. Schimbă culoarea apei: utilizatorul poate alege schimbarea culorii obstacolelor înainte de începerea execuției animației;
4. Salvare imagine: salvează într-un fișier de tip bitmap, jpg sau png zona matricei;
5. Încărcare exemplu: se alege un fișier de intrare de tip .in sau .txt;
6. Exportă imaginea ca text: matricea de butoane este transformată în numere, fiecare număr reprezentând o zonă, iar cu *-1* sunt marcate obstacolele.
7. Cea mai mare zonă: afișează cea mai mare zonă din matrice



1. Fila *Solutie C++*

În această filă utilizatorului i se afișează soluția în C++.



# COD SURSĂ

private void button1\_Click(object sender, EventArgs e)

{

panel1.Controls.Clear();

button3.Enabled = true;

button4.Enabled = true;

button5.Enabled = true;

button6.Enabled = true;

checkBox1.Enabled = true;

timer1.Stop();

fill = false;

v = 0;

checkBox1.Enabled = true;

n = Convert.ToInt32(numericUpDown1.Value);

m = Convert.ToInt32(numericUpDown2.Value);

generare();

for (int i = 0; i < n; i++)

{

for (int j = 0; j < m; j++)

{

B[i, j] = new Button();

B[i, j].FlatStyle = FlatStyle.Flat;

B[i, j].Width = B[i, j].Height = L;

B[i, j].Top = i \* L;

B[i, j].Left = j \* L;

B[i, j].Click += new EventHandler(click);

panel1.Controls.Add(B[i, j]);

}

}

if (button5.Bottom < panel1.Height)

this.Height = panel1.Height + L;

else this.Height = button4.Bottom + L;

this.Width = panel1.Right + L;

}

private void button2\_Click(object sender, EventArgs e)

{

checkBox1.Enabled = false;

button3.Enabled = false;

button4.Enabled = false;

button5.Enabled = false;

button6.Enabled = false;

fill = true;

bool ok = false;

Q = new Queue<pozitie>();

for (int i = 0; i < n && !ok; i++)

for (int j = 0; j < m; j++)

{

if (A[i, j] == 0)

{

pozitie a;

a.x = i;

a.y = j;

Q.Enqueue(a);

v++;

Cnt[v]++;

B[i, j].BackColor = C[v];

A[i, j] = v;

B[i, j].Text = A[i, j].ToString();

ok = true;

timer1.Start();

break;

}

}

}

private void timer1\_Tick(object sender, EventArgs e)

{

if (Q.Count != 0)

{

pozitie Coord = Q.Peek();

for (int k = 0; k < 4; k++)

{

iv = Coord.x + di[k];

jv = Coord.y + dj[k];

if (iv >= 0 && iv < n && jv >= 0 && jv < m && A[iv, jv] != -1 && A[iv, jv] != v)

{

A[iv, jv] = v;

B[iv, jv].Text = A[iv, jv].ToString();

B[iv, jv].BackColor = C[v];

Cnt[v]++;

pozitie a;

a.x = iv;

a.y = jv;

Q.Enqueue(a);

}

else if (iv >= 0 && iv < n && jv >= 0 && jv < m)

{

Color c = B[iv, jv].BackColor;

B[iv, jv].BackColor = Color.Red;

B[iv, jv].Refresh();

System.Threading.Thread.Sleep(500);

B[iv, jv].BackColor = c;

}

}

Q.Dequeue();

}

else

{

bool ok = false;

for (int i = 0; i < n && !ok; i++)

for (int j = 0; j < m; j++)

if (A[i, j] == 0)

{

pozitie a;

a.x = i;

a.y = j;

Q.Enqueue(a);

v++;

Cnt[v]++;

A[i, j] = v;

B[i, j].BackColor = C[v];

B[i, j].Text = A[i, j].ToString();

ok = true;

break;

}

if (!ok)

{

timer1.Stop();

MessageBox.Show("In matrice sunt " + v.ToString() + " zone");

button3.Enabled = true;

button4.Enabled = true;

button5.Enabled = true;

button6.Enabled = true;

}

}

}

private void click(object sender, EventArgs e)

{

Button a = (Button)sender;

if (fill == true)

return;

if (a.BackColor == delimiter)

{

a.BackColor = Color.Empty;

A[a.Top / L, a.Left / L] = 0;

}

else

{

a.BackColor = delimiter;

A[a.Top / L, a.Left / L] = -1;

}

}

# BIBLIOGRAFIE

1. [www.pbinfo.ro](http://www.pbinfo.ro)
2. [www.stackoverflow.com](http://www.stackoverflow.com)